|  |  |  |
| --- | --- | --- |
| **Selection Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 499500 | 0. 036777496337890625 |
| **2,000 (observed)** | 1999000 | 0. 14583635330200195 |
| **4,000 (observed)** | 7998000 | 0. 5218839645385742 |
| **8,000 (observed)** | 31996000 | 2. 153265953063965 |
| **16,000 (observed)** | 127992000 | 10. 081698179244995 |
| **32,000 (observed)** | 511984000 | 37. 488874435424805 |
| **100,000 (estimated)** | 5000050000 | 366.1173916 |
| **500,000 (estimated)** | 125000250000 | 9152.861567 |
| **1,000,000 (estimated)** | 500005000000 | 36611.73916 |
| **10,000,000 (estimated)** | 50000005000000 | 3661173.916 |

|  |  |  |
| --- | --- | --- |
| **Insertion Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 247986 | 0. 04264044761657715 |
| **2,000 (observed)** | 1018717 | 0.13101649284362793 |
| **4,000 (observed)** | 3995264 | 0. 5313568115234375 |
| **8,000 (observed)** | 16112194 | 2. 2796738147735596 |
| **16,000 (observed)** | 64667449 | 8. 63548731803894 |
| **32,000 (observed)** | 257507119 | 34. 02519655227661 |
| **100,000 (estimated)** | 2.499975 x 10^9 | 330.3292782 |
| **500,000 (estimated)** | 6.2499875 x 10^10 | 8258.298022 |
| **1,000,000 (estimated)** | 2.4999975 x 10^11 | 33033.22512 |
| **10,000,000 (estimated)** | 2.49999975 x 10^13 | 3303325.485 |

1. Which sort do you think is better? Why?

I think that selection sort is better than insertion sort when it comes to smaller test cases; however, insertion sort runs faster than selections sort when it comes to larger test cases this is because selection sort has to iterate through a larger and larger list before making a single swap which is not so much of a problem if it is a smaller list, but it is for larger lists.

1. Which sort is better when sorting a list that is already sorted (or mostly sorted)? Why?

Insertion sort is immensely better at sorting already sorted (or mostly sorted) lists because insertion utilizes checks and if the check passes, which will most likely happen because it is mostly sorted it will continue to traverse through the list which gives us a O(n) time complexity if it is already sorted, while selection sort has to iterate through the whole list to make one swap whether it is sorted or not, all cases being an O(n^2) time complexity.

1. You probably found that insertion sort had about half as many comparisons as selection sort. Why? Why are the times for insertion sort not half what they are for selection sort? (For part of the answer, think about what insertion sort has to do more of compared to selection sort.)

Insertion sort had about half as many comparisons as selection sort because insertion sort does not need to compare the whole list to make one swap like selection sort. Insertion sort only compares the next item and until the item is properly sorted. Therefore, in the worst case insertion sort would have to compare an average of half of the length of the list compared to the whole list selection sort must traverse through. Nonetheless, insertion sort does not run about half the time compared to selection sort because most sample tests are not already sorted; therefore, it cannot take advantage of the trait of not making a comparison for each swap, making the average case for both insertion and selection sort, O(n^2).

|  |  |  |
| --- | --- | --- |
|  | **Number of Quicksort Comparisons** | |
| **Starting List** | pivot = first | pivot = median of 3 |
| Ordered, ascending |  |  |
| n = 100 | 4950 | 2500 |
| n = 200 | 19900 | 10000 |
| n = 400 | 79800 | 40000 |
| n = 800 | 319600 | 16000 |
| Random |  |  |
| n = 100 (average 10 runs) | 682.4 | 643.6 |
| n = 200 (average 10 runs) | 1480.1 | 1601.1 |
| n = 400 (average 10 runs) | 3561.2 | 3581.0 |
| n = 800 (average 10 runs) | 8470.2 | 8404.3 |
|  |  |  |
| Observed Big O() behavior, ordered with pivot = first : O(nlog n) | | |
| Observed Big O() behavior, ordered with pivot = median of 3 : O(nlog n) | | |
| Observed Big O() behavior, random with pivot = first : O(nlog n) to O(n^2) | | |
| Observed Big O() behavior, random with pivot = median of 3 : O(nlog n ) | | |
| For random list, observation regarding using first vs. median of 3 : There is no clear method that runs faster or slower than the other. | | |